CLAIMS

We claim:

1. Dynamic verification of the validity of computer-executable instructions.

2. In a computer that includes at least one processor that executes instructions stored in a memory, which is organized into separately addressable memory blocks, a method for verifying the validity of instructions comprising:

for at least one current instruction that has been identified for submission to the processor for execution, determining an identifying value for a current memory block that contains the current instruction;

comparing the identifying value of the current memory block with a set of reference values;

if the identifying value satisfies a validation condition, allowing execution of the current instruction by the processor; and

if the identifying value does not satisfy the validation condition, generating a response;

whereby the current instruction is verified dynamically before being executed.

3. A method as in claim 2, further comprising:

including in the set of reference values at least one validation entry corresponding to at least one identifying value for predetermined contents of a known, valid memory block;

in which the validation condition is that the identifying value of the current memory block matches any validation entry in the set of reference values.

4. A method as in claim 2, further comprising:

including in the set of reference values at least one invalidation entry corresponding to at least one identifying value for predetermined contents of a known invalid memory block;

in which the validation condition is that the identifying value of the current instruction differs from all invalidation entries in the set of reference values.

5. A method as in claim 2, in which:

the step of determining the identifying value of the current memory block comprises computing a hash value as a function of at least a sub-set of the contents of the current memory block; and

each reference value is computed as a hash value of at least a sub-set of a known, reference memory block.

6. A method as in claim 5, in which the step of computing the hash value of both the current memory block and at least one of the reference memory blocks comprises computing the respective hash values based on only part of the contents of both the current and the reference memory blocks.

7. A method as in claim 6, further comprising:

identifying, in at least one reference memory block, non-indicative contents that are valid but that are at least potentially non-constant such that they do not indicate validity of the reference memory block as a whole;

before or when computing the hash value for the reference memory block, applying a mask to the contents of the reference memory block such that the non-indicative contents do not influence the computed hash value; and

before or when computing the hash value for the current memory block, applying the mask to the current memory block contents.

8. A method as in claim 2, further comprising:

for each of a plurality of memory blocks, indicating in a structure whether each respective block is validated; and

for each current instruction from a memory block whose structure indication is that it is validated, directly allowing execution of the current instruction.

9. A method as in claim 8, in which the step of indicating in the structure whether the plurality of memory blocks is validated comprises causing a corresponding entry to be made in a group of hardware attribute indicators.

10. A method as in claim 9, in which the hardware attribute indicators are execute and write permission attributes associated with an entry in a translation lookaside buffer.

11. A method as in claim 8, in which the step of indicating in the structure whether the plurality of memory blocks is validated comprises making a corresponding entry in a software data structure.

12. A method as in claim 8, further comprising:

performing the steps of determining the identifying value for the current memory block and comparing the identifying value of the current memory block with a set of reference values only for current instructions located in memory blocks not indicated in the structure as being validated; and

if the identifying value of a current memory block not indicated as being validated satisfies the validation condition, then setting the corresponding structure indication to indicate that it is validated.

13. A method as in claim 12, further comprising:

sensing modification of any memory block for which the structure includes an indication and, upon sensing modification of any such memory block, setting its indication in the structure to indicate that the memory block is not validated.

14. A method as in claim 8, further comprising:

determining a branch history for the current instruction; and

checking whether the memory blocks in which instructions in the branch history are located are validated, the validation condition including the requirement that each checked memory block in the branch history is validated.

15. A method as in claim 2, further comprising performing the steps of determining the identifying values for current memory blocks, comparing the identifying values with the set of reference values, and determining whether the validation condition has been satisfied only after the occurrence of a triggering event.

16. A method as in claim 15, in which the triggering event is the writing of at least one new unit of code or data to any physical component within the computer.

17. A method as in claim 15, in which the triggering event is the attempted execution of any instruction located on any unverified memory block.

18. A method as in claim 15, in which the triggering event is the attempted execution of any instruction located on any unverified memory block of newly installed software.

19. A method as in claim 15, further comprising triggering dynamic verification depending on the identity of the user of the computer who has caused submission of the current instruction.

20. A method as in claim 15, further comprising triggering dynamic verification depending on a context in which the current instruction is submitted for execution.

21. A method as in claim 20, in which the context is a level of security clearance associated with the computer, a user of the computer, or a program of which the current instruction is a part.

22. A method as in claim 2, further comprising verifying only a sample of the current instructions.

23. A method as in claim 22, in which the sample is a time-sampled sub-set of current instructions.

24. A method as in claim 22, in which the sample is a sequentially sampled sub-set of current instructions.

25. A method as in claim 22, in which the sample is a sub-set of current instructions sampled spatially, over a range of addresses or equivalent memory block identifiers.

26. A method as in claim 2, in which the step of generating the response comprises terminating a software entity with which the current memory block is associated.

27. A method as in claim 2, in which the step of generating the response comprises suspending execution of a software entity with which the current memory block is associated.

28. A method as in claim 2, in which the step of generating the response comprises posting a message to a user, system administrator, or other predetermined recipient.

29. In a computer that includes a virtual machine running in a direct execution mode on an underlying hardware platform via an intermediate software layer, the method as in claim 2, in which the step of generating the response comprises switching the execution mode of the virtual machine to binary translation.

30. In a computer that includes a virtual machine running on an underlying hardware platform via an intermediate software layer, the method as in claim 2, in which the step of generating the response includes checkpointing the state of the virtual machine.

31. A method as in claim 2, further comprising:

associating different responses with at least two different memory blocks;

upon detection of failure of the current instruction to satisfy the validation condition, generating the response associated with the memory block in which the current instruction is located.

32. A method as in claim 2, further comprising:

tracking which programs are being executed within the computer by associating the reference values with respective predetermined programs, a match between the identifying value of the current memory block with any validation entry in the set of reference values indicating execution of the corresponding one of the predetermined programs.

33. In a computer that includes a virtual machine (VM) running on an underlying hardware platform via an intermediate software layer operable to switch the virtual machine between a direct execution mode and a binary translation mode, the method as in claim 2, further comprising verifying the validity of VM-issued instructions in conjunction with binary translation of any of the VM-issued instructions.

34. In a computer that includes at least one processor that executes instructions stored in a memory, which is organized into separately addressable memory blocks, a method for verifying the validity of instructions comprising:

computing a set of at least one reference value as a hash value of at least a sub-set of a known, reference memory block;

upon occurrence of a triggering event, for at least one current instruction that has been identified for submission to the processor for execution, determining an identifying value for a current memory block that contains the current instruction by computing a hash value as a function of at least a sub-set of the contents of a current memory block;

comparing the identifying value of the current memory block with the set of reference values;

if the identifying value satisfies a validation condition, allowing execution of the current instruction by the processor; and

if the identifying value does not satisfy the validation condition, generating a response;

including in the set of reference values at least one validation entry corresponding to at least one identifying value for predetermined contents of a known, valid memory block,

the validation condition being that the identifying value of the current memory block matches any validation entry in the set of reference values;

whereby the current instruction is verified dynamically before being executed.

35. A system for verifying the validity of executable code in a computer comprising:

at least one processor;

a mechanism for identifying at least one current instruction that has been identified for submission to the processor for execution;

a memory that is organized into separately addressable memory blocks, the

a verification engine comprising computer-executable code

for at least one current instruction that has been identified for submission to the processor for execution, for determining an identifying value for a current memory block that contains the current instruction;

for comparing the identifying value of the current memory block with a set of reference values;

if the identifying value satisfies a validation condition, for allowing execution of the current instruction by the processor; and

if the identifying value does not satisfy the validation condition, for generating a response;

whereby the current instruction is verified dynamically before being executed.

36. A system as in claim 35, further comprising:

at least one validation entry included in the set of reference values corresponding to at least one identifying value for predetermined contents of a known, valid memory block;

in which the validation condition is that the identifying value of the current memory block matches any validation entry in the set of reference values.

37. A system as in claim 35, further comprising:

at least one invalidation entry included in the set of reference values corresponding to at least one identifying value for predetermined contents of a known invalid memory block;

in which the validation condition is that the identifying value of the current instruction differs from all invalidation entries in the set of reference values.

38. A system as in claim 35, further including:

a hashing module within the verification engine comprising computer-executable code for determining the identifying value of the current memory block by computing a hash value as a function of at least a sub-set of the contents of the current memory block; and

for computing each reference value as a hash value of at least a sub-set of a known, reference memory block.

39. A system as in claim 38, further comprising a sub-set selection structure for selecting only a sub-set of the current memory block for computation of the respective hash value.

40. A system as in claim 39, in which the sub-set selection structure is a mask.

41. A system as in claim 35, further comprising a structure containing an indication, for each of a plurality of memory blocks, of whether each respective block is validated, the verification engine being further provided with computer-executable code for directly allowing execution of the current instruction for each current instruction from a memory block whose structure indication is that it is validated.

42. A system as in claim 41, in which the structure containing the indications is a group of hardware attribute indicators.

43. A system as in claim 42, in which the hardware attribute indicators are execute and write permission attributes associated with an entry in a translation lookaside buffer.

44. A system as in claim 35, further comprising a software module comprising computer-executable instructions for selecting for verification only a sample of the current instructions.

45. A system as in claim 35, further comprising:

a hardware platform that includes the processor(s);

an intermediate virtualization layer;

a virtual machine (VM) running in a direct execution mode or a binary translation mode on the underlying hardware platform via the intermediate virtualization layer;

the intermediate virtualization layer being provided for switching the execution mode of the virtual machine to binary translation.

46. A system as in claim 45, in which the verification engine is further provided for verifying the validity of VM-issued instructions in conjunction with binary translation of any of the VM-issued instructions.

47. A system as in claim 45, in which the verification engine is further provided for triggering the intermediate virtualization layer to switch execution of the virtual machine to the binary translation mode as the response.